IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application		)	PATENT APPLICATION
Inventors:	Cirne, et al.	)	
Application No.:	10/700,338	)	Art Unit: 2192
Filed:	November 3, 2003	)	Examiner: Wei, Zheng
Title:	SIMPLE METHOD OPTIMIZATION	)	Customer No. 28554

APPEAL BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This brief is submitted in accordance with 37 C.F.R. §41.37, following the Notice of Appeal filed by Appellants on January 9, 2008. The fee set forth in 1.17(c) is submitted herewith.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST *(37 C.F.R. §41.37(c)(i))*

The real party in interest is Computer Associates Think, Inc., the assignee of record.

II. RELATED APPEALS AND INTERFERENCES *(37 C.F.R. §41.37(c)(ii))*

Appellants know of no other appeals or interferences which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS *(37 C.F.R. §41.37(c)(iii))*

Claims 1-3, 5-15, 17-24, 26-35, 37-42, and 44-52 are pending in this application.

Claims 1-3, 5-15, 17-24, 26-35, 37-42, and 44-50 stand finally rejected and are the subject of this appeal.

New claims 51 and 52 contain no new matter and were added after final rejection of the claims, but prior to the receipt of an Advisory Action, in order to place the application in better form for appeal pursuant to 37 C.F.R. §1.116(b). However, the Examiner has not entered new claims 51 and 52.

Claims 4, 16, 25, 36, and 43 have been cancelled.

Appellants herein appeal from the final rejection of claims 1-3, 5-15, 17-24, 26-35, 37-42, and 44-50 and respectfully request consideration of new claims 51 and 52 for appeal, as claims 51 and 52 place the application in better form for appeal. The claims as currently pending are reproduced in the Claims Appendix.

IV. STATUS OF AMENDMENTS *(37 C.F.R. §41.37(c)(iv))*

In response to the final Office Action dated July, 12, 2007, Appellants submitted amendments to claims 49 and 50 and added new claims 51 and 52 in order to place the application in better form for appeal pursuant to 37 C.F.R. §1.116(b). These amendments were submitted in the Response to the Final Office Action dated September 5, 2007. However, in the Advisory Action from the Examiner dated December 27, 2007, the Examiner indicated that the amendments were not entered because the amendments required further consideration. Applicants respectfully disagree with the Examiner's decision to deny entry of the amendments for the reasons explained in the Argument section.

Claim 49 was amended to include the limitation that "said method is non-synthetic." The amended claim contains features similar to claim 5. Claim 49 (with the proposed amendment shown) is reproduced in the Claims Appendix.

Claim 50 was amended to change the claim's dependency from claim 47 to claim 49. The amended claim contains features similar to claim 17. Claim 50 (with the proposed amendment shown) is reproduced in the Claims Appendix.

New claims 51 and 52 were added to place the application in better form for appeal by reciting the limitations of method claim 17 in claims for an apparatus. Apparatus claims 51 and 52 are reproduced in the Claims Appendix.

Any and all other amendments made to claims 1-3, 5-15, 17-24, 26-35, 37-42, and 44-48 to date have been entered.

V. SUMMARY OF CLAIMED SUBJECT MATTER *(37 C.F.R. §41.37(c)(v))*

The technology recited in claims 1-3, 5-15, 17-24, 26-35, 37-42, and 44-52 generally relate to "technology for monitoring applications" (Specification p. 2, lines 16-17). As discussed in the Background of the Invention section, prior monitoring techniques, such as the implementation of a performance analysis tool, provide an indication of an application's performance. These indications may, for example, "provide timing data on how long each method (or procedure or other process) is being executed, report how many times each method is executed and/or identify the function call architecture" (Specification p. 1, lines 22-25). However, many times these tools provide too much data, which problematically creates for the user the "difficult task of analyzing the multitude of data to determine which data is relevant and which data is not relevant (e.g. redundant or otherwise not useful)" (Specification p. 2, lines 3-4). In many instances, prior monitoring techniques would "instrument a large number of methods in the software in order to be able to analyze the performance of each method. However, modifying a large number of methods… [adds] an enormous amount of code to the software and may impact performance of the underlying software" (Specification p. 2, lines 11-13).

The technology recited in the claims for the present application resolves these issues by only modifying certain methods for purposes such as monitoring or tracing, making the process of monitoring or tracing applications much more efficient and useful. Specifically, the technology of the present application modifies methods that are determined to be complex. Methods that are determined to be simple are not modified. As stated in the Specification:

> In one embodiment, a method is complex if it meets three criteria: (1) the method has an access level of public or package; (2) the method is non-synthetic and (3) the method calls at least one other method. Methods that do not satisfy all three criteria are classified as simple methods. In other embodiments, a method can be classified as complex if it satisfies two of the above criteria, or other similar criteria (p. 12, lines 17-23).

Software typically contains methods that call other methods. For example, in Figure 4 of the Drawings, method M1 calls methods M2 and M3, and method M3 calls methods M4 and M5, both

of which do not call another method (also see Specification p. 11, line 23 – p. 12, line 11). Monitoring M1 would result in the monitoring of all of the methods that it calls. It would be inefficient and redundant to also monitor the methods that it calls separately. Therefore, it is useful to monitor only methods which call at least one other methods.

The Specification describes a method's access level as follows:

> Java provides for four levels of access control for methods: public, private, protected, and package. Private is the most restrictive access level. A private method can only be accessed by methods in the same class. A protected method can be accessed by other methods in the same class, sub classes and classes in the same package. A public method can be accessed by any class of any parentage in any package. The package access level is the default and allows a method to be accessed by classes in the same package, regardless of their parentage" (p. 12, line 24 – p. 13, line 5).

In the example code found on page 13, line 13 to page 14, line 9 of the Specification, the access level can be determined by referencing the access level indicated prior to each method name (e.g. public methodOne()).

A method is determined to be synthetic if it is "a method that does not appear in the source code. For example, during compilation, the compiler may add one or more methods. Typically, the compiler explicitly makes it easy to see that methods are or are not synthetic. Java compilers flag these methods with the 'Synthetic' attribute" (Specification p. 13, lines 6-11).

As shown in Figure 5 of the Drawings, "various methods are accessed and determined to be either complex or simple… If the method is complex, it is modified… If the method is determined to be simple, it is not modified for the purposes that the complex methods are modified" (Specification p. 14, lines 19-23). Figure 6 of the Drawings shows one way a method that is determined to be complex can be modified. A method is modified by accessing the beginning of the byte code for the method (step 302), adjusting the byte code indices to prepare for the addition of code (step 304), adding new start byte code (step 306), accessing the end of the byte code (step 308), adding new exit byte code (step 310), adjusting the exception table due to the adjustment of the byte code indices (step 312), and adding a new exception table entry in the exception table (step 314) (also see

Specification p. 15, line 14 – p. 16, line 21). Such modification to the code for a method can be performed for any purpose, such as monitoring or tracing the method.

Figure 3 of the Drawings describes a process for monitoring using a modified method, which includes receiving existing code (step 260), receiving new function(s) such as "new classes and methods that allow for monitoring the application" (step 262), modifying existing code (as described in Figure 6, for example) (step 264), adding "all or part of the new functionality (e.g. the new classes/methods)… [to] the existing code" (step 266), storing the modified code (step 268), and running the modified code for monitoring (step 270) (Specification p. 10, lines 10-21). The Specification describes that the process for monitoring may be implemented, for example, through "software that is stored on one or more processor readable storage devices [that] is used to program one or more processors (p. 10, lines 1-2).

Claim 39 recites means plus function language for an "apparatus capable of monitoring." The claim recites "means for determining whether a method calls another method." The Specification discloses that the "means for determining" can be "implemented in software that is stored on one or more processor readable storage devices… to program one or more processors" (p. 9, line 27 – p. 10, line 2). The "means for determining" can further be incorporated in "computing devices [which] will include [the] one or more processor in communication with one or more processor readable storage devices" (p. 9, lines 18-20). The determining whether a method calls another method is discussed and shown in Figure 5 of the Drawings in the decision block of step 286. "If the method under consideration does not call any other methods, then that method is not modified for the purposes that the complex methods are modified (step 290). If the method under consideration does call one or more methods, then the method under consideration is modified" (Specification p. 15, lines 6-8).

Claim 39 further recites "means for tracing said method for a particular purpose only if said method calls another method." The "means for tracing" can be "implemented in software that is stored on one or more processor readable storage devices," much like the "means for determining" (p. 9, line 27 – p. 10, line 2). Tracing the method is discussed and shown in step 288 of Figure 5 of

the Drawings. Figure 5 of the Drawings shows that if the method is determined to call other methods (step 286), the method will be modified (step 288) "to add the tracer" (Specification p. 15, line 9). "If the method under consideration does not call any other methods, then that method is not modified for the purposes that the complex methods are modified," such as to add the tracer (Specification p. 15, lines 6-9). The tracing mechanism is used when the modified code is run (step 270 of Figure 3 of the Drawings).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL *(37 C.F.R. §41.37(c)(vi))*

A. Whether amendments to claims 49-52 have been properly denied entry under 37 C.F.R. §1.116 because the claims require further consideration.

B. Whether claims 1, 7, 8, 10, 12, 13, 20, and 39 have been properly rejected under 35 U.S.C. §102(e) as being anticipated by Berkley (US 6,351,843).

C. Whether claims 2, 3, 5, 6, 14, 15, 17, 18, 40-42, 44, and 47-50 have been properly rejected under 35 U.S.C. §103(a) as being unpatentable over Berkley.

D. Whether claims 9, 11, 19, 21-24, 26-35, 37, 38, 45, and 46 have been properly rejected under 35 U.S.C. §103(a) as being unpatentable over Berkley in view of Berry (US 6,662,359).

VII. ARGUMENT *(37 C.F.R. §41.37(c)(vii))*

A. Denial of Entry of Amendments for Claims 49-52 Under 37 C.F.R. §1.116

Amendments to claims 49-52 have been denied entry by the Examiner under 37 C.F.R. §1.116. Applicants request that the claims be entered because the claims place the application in better form for appeal pursuant to 37 C.F.R. §1.116, which states:

> (b) After a final rejection… in an application…, but before or on the same date of filing an appeal…:
>
> . . .
>
> (2) An amendment presenting rejected claims in better form for consideration on appeal may be admitted…

The amendments to claims 49-52 were submitted in the Response to the Final Office Action dated September, 5, 2007. These amendments were submitted prior to receipt of the Advisory Action dated December 27, 2007 and prior to filing the Notice of Appeal filed on January 9, 2008. Applicants respectfully assert that claims 49-52 do not contain any new matter because they merely recite limitations from previously entered claims.

Applicants respectfully request that the amendments to claims 49 and 50 be entered because they place the application in better form for appeal by simplifying the issues on appeal. The amendments made to claims 49 and 50 were made to incorporate all three criteria for classifying a method as complex (i.e. a method that "calls another method," "is non-synthetic," and "has an access level of public or package"). The features of the amended claims are similar to those recited in method claim 5; however, claims 49 and 50 further clarify what the term "complex" encompasses in the claims.

Additionally, Applicants assert that new claims 51 and 52 be entered for at least the same reasons as claims 49 and 50. Claim 51 was added to further clarify how the method is modified (i.e. "adding a tracer"). Claim 52 was added to further clarify which methods should be modified.

As will be discussed in more detail below, the Examiner believes the limitations of

determining whether a method "calls another method," "is non-synthetic," and "has an access level of public or package" is disclosed in the prior art. Applicants respectfully disagree with the Examiner because the prior art is not capable of determining any one of these criteria, as will be discussed in more detail below. Applicants respectfully assert that the proposed amendments to claims 49-52 place the application in better form for appeal because the amendments simplify this issue on appeal by clarifying the difference between the present application and the prior art.

Moreover, the Examiner argues that the amendments should not be entered because the amendments would require further consideration (see Advisory Action). However, the Examiner does not state why he believes the amendments would require further consideration. MPEP §706.07(f) states that if a "proposed amendment presents new issues requiring further consideration and/or search, the examiner should provide an explanation as to the reasons why the proposed amendment raises new issues that would require further consideration and/or search." The Examiner has not provided such explanation.

Therefore, Applicants respectfully request that the amendments to claims 49-52 be entered pursuant to 37 C.F.R. §1.116 because the claims place the application in better form for appeal. Applicants respectfully request that the amended claims be entered.

B. Rejection of Claims 1, 7, 8, 10, 12, 13, 20, and 39 Under 35 U.S.C. §102(e)

Claims 1, 7, 8, 10, 12, 13, 20, and 39 have been rejected under 35 U.S.C. §102(e) as being anticipated by Berkley. Applicants respectfully traverse the rejection because Berkley does not disclose each and every limitation recited in claims 1, 7, 8, 10, 12, 13, 20, and 39.

Berkley discloses monitoring classes in an object-oriented environment. A class may include one or more objects, and the "class defines how an object is implemented. Objects are instances of classes" (Berkley col. 4, lines 61-62). Each object may include one or more methods, as shown in Figure 1. Therefore, a class may contain one or more objects, which each may contain one or more methods. As shown in Figure 1, a method may or may not call another method. For example, in

Object 1, which is part of a class, Method A calls Method B. However, Method D does not call any other method.

In Berkley, methods are monitored by class. For example, if a user wishes to monitor a particular method within a program, the user can change configuration settings associated with the program to specify or activate the class of the method that the user wishes to monitor. "[C]onfiguration settings 300 received as part of a program's execution code are modified at runtime to add a setting to specify (as one example) tracing for a desired class of the executable 310, thereby producing new configuration settings 320" (Berkley Figure 5 and col. 6, lines 52-53). Berkley further explains that:

> the new configuration settings 330 are then employed with the existing application executable 340 to run the application executable 350. The object runtime will query the configuration settings and when a trace is active for a given class, will dynamically insert the trace class into the inheritance hierarchy for that class… Runtime will then see the trace class within the hierarchy and setup for a trace 360 (Berkley col. 6, line 64 – col. 7, line 4).

For example, for a class named 'class 1' that is specified in the configuration settings, the trace is set up through "redirection stubs [that] are created [to] trace entry and exit methods around each target method within class 1" (Berkley col. 7, lines 61-63).

A trace is active for a given class when the user specifies the class in the configuration settings. The example for activating classes in column 7, lines 20-26 of Berkley shows:

> configuration settings used to set up a desired tracing environment. A configuration variable is initialized with the name(s) of the class(es) whose methods should be traced:
> [TraceOptions]
> ClassTrace = Class1, Class2, Class3, Class4

In the above example, Class1, Class2, Class3, and Class4 are set active by specifying these classes in the ClassTrace list. All of the methods in these classes will be traced. Using Figure 1 of Berkley as an example, if Object 1 was contained within Class 1, which is specified in the

ClassTrace list above, all of the methods contained within Object 1 would be modified for tracing, even Method D which does not call any other method.

Claim 1 is not anticipated by Berkley because Berkley does not disclose "determining whether to modify said method, said step of determining whether to modify said method includes determining whether said method calls another method." Instead, Berkley "determines whether to modify said method" if the class for the method is specified in the ClassTrace list of the configuration settings. On page 4 of the final Office Action dated July 12, 2007, the Examiner argues that this step of "determining" is disclosed in Figure 1 of Berkley, because Figure 1 shows methods that call other methods. The Examiner makes the same arguments in the Response to Arguments section on page 3 of the final Office Action. Although it is true that Figure 1 shows methods that "call another method," the determination of "whether to modify said method" is not made based on "whether said method calls another method." Instead, the decision to "modify said method" is based on "configuration settings [that] are checked to determine whether the class… is to be traced, i.e., is class 1 in a ClassTrace list defined by the user" (Berkley col. 7, lines 46-49).

Furthermore, on page 4 of the final Office Action, the Examiner argues that "determining whether [a] function is active for a class of the executable" (Berkley col. 2, lines 39-40) can be equated to "determining whether said method calls another method," as recited in claim 1. The Examiner makes the same arguments in the Response to Arguments section on page 3 of the final Office Action. However, this interpretation of Berkley is incorrect. A function is active if it is specified by a user in the ClassTrace list. The determination of whether to "modify said method" is made by checking this list and has nothing to do with whether a "method calls another method." Therefore, Berkley does not disclose this feature recited in claim 1.

Additionally, in the Advisory Action dated December 27, 2007, the Examiner makes the same argument he made in the final Office Action. More specifically, on page 2 of the Advisory Action, the Examiner states that as a whole, Berkley discloses tracing by modifying methods through the addition of macro calls (Berkley col. 1, line 62 – col. 2, line 6). In this referenced passage, Berkley states the following:

> Existing object-oriented tracing methods typically employ macro calls that must be physically added to the code by a programmer, this leave much room for error and it also means that the method that is not physically modified will not be traced. Further, such macro calls, once inserted, are in the code and must be executed even if one is not interested in tracing. Even with a trace flag check, there will be millions of wasted instructions. Since every routine has the same two macros, tracing will only have an on/off quality which allows the user to turn tracing on for every method or off for every method.

The Examiner argues that this passage discloses that "in order to add 'macro calls ' to a specific method, 'whether said method calls another method' has to be determined" (Advisory Action, p. 2). It is unclear how the Examiner derived this conclusion from this passage from Berkley. From this passage, it appears that no such determination is required, and any method may be modified through macro calls, regardless of whether the "method calls another method." Therefore, claim 1 is not anticipated by Berkley.

In the Advisory Action, the Examiner further argues that Berkley discloses "determining whether said method calls another method" because Figure 7 in Berkley shows that "the 'Method 1' called by the application's function/method, has to be determined first and then combined with configuration settings to insert 'Redirection Stub 1' in application's caller function/method to further trace/monitor the application. Therefore, Berkley does disclose the cited limitation" (Advisory Action, p. 2). Applicants respectfully disagree that Figure 7 discloses the features of claim 1. Figure 7 shows a modified method (e.g. "Method 1") with the added redirection stub. If an application's function/method calls "Method 1," there is no determination made as to whether "Method 1" "calls another method." Instead, the application will check the configuration settings to see if "Method 1" is active. If the method is active, the redirection stub is inserted in the code. In Figure 7, "when a program invokes method 1, it will not be invoked directly. Rather, the redirection stub is inserted before method 1, and is inclusive of method 1" (Berkley col. 8, lines 50-53). The program would not need to determine whether "Method 1" "calls another method." Therefore, Applicants respectfully assert that claim 1 is not anticipated by Berkley.

For at least the same reasons stated above, Berkley also does not disclose "modifying said method for a particular purpose if said method calls another method," as recited in claim 1. As previously discussed, Berkley discloses "modifying said method" in an executable if the method is "active for a class of the executable" (Berkley col. 2, lines 39-40). It does not matter whether the "method calls another method." Any method within a class that is specified in the ClassTrace list will be modified, even those methods within the class that do not "call another method." Therefore, claim 1 is not anticipated by Berkley because Berkley does not disclose "modifying... if said method calls another method."

Claims 7, 8, 10, 12, 13, 20, and 39 each recite limitations similar to those recited in claim 1. Therefore, claims 7, 8, 10, 12, 13, 20, and 39 are also not anticipated by Berkley for at least the same reasons stated for claim 1. Applicants respectfully assert that claims 1, 7, 8, 10, 12, 13, 20, and 39 are allowable over the cited prior art. Based on the above, it is respectfully requested that the rejection of claims 1, 7, 8, 10, 12, 13, 20, and 39 under 35 U.S.C. §102(e) be withdrawn.

C. Rejection of Claims 2, 3, 5, 6, 14, 15, 17, 18, 40-42, 44, and 47-50 Under 35 U.S.C. §103(a)

Claims 2, 3, 5, 6, 14, 15, 17, 18, 40-42, 44, and 47-50 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Berkley. Applicants respectfully traverse the rejection as follows.

1. Claims 6, 18, 40, 44, and 47

Applicants respectfully assert that claims 6, 18, 40, 44, and 47 are not obvious over Berkley because Berkley does not disclose, teach, or suggest all of the limitations of the rejected claims. Specifically, Berkley does not disclose, teach, or suggest "determining whether to modify said method, said step of determining whether to modify said method includes determining whether said method calls another method" and "modifying said method for a particular purpose if said method

calls another method," as recited in claim 1. Claims 6, 18, 40, 44, and 47 each contain similar features. Applicants respectfully assert that the claims are in condition for allowance.

As discussed above, Berkley discloses modifying a method if the class for that method is specified in a ClassTrace list (Berkley col. 7, lines 46-49). The decision to modify a method is only based on whether the class for the method is listed in the ClassTrace list; the decision has nothing to do with whether the "method calls another method." Furthermore, it would not be obvious to one of ordinary skill in the art to develop the claimed features. The knowledge of one having ordinary skill in the art adds nothing regarding these features because Berkley can only modify all of the methods within a class. The modification is performed by "determining whether the class is… in a Class Trace list defined by the user… [and if] the class is to be traced, …tracing is begun by allocating storage for a new class hierarchy table… [The] trace class is inserted into one of the entries in this new class hierarchy table" (Berkley col. 7, lines 46-56). The new class hierarchy table is then used to execute the tracing feature. It would not be obvious to modify Berkley to include the feature of "determining" and "modifying… if said method calls another method" because Berkley cannot modify specific methods within a class. The entire class must be modified by creating a new class hierarchy table for the entire class, regardless of whether a method in the class "calls another method." One of ordinary skill in the art could not modify Berkley to include this feature because Berkley cannot be modified to select particular methods to trace based on criteria, such as "whether said method calls another method," since the entire class hierarchy table must be modified for all methods in the class. Therefore, Berkley would not lead one of ordinary skill in the art to develop the claimed features as recited in claims 6, 18, 40, 44, and 47. Applicants respectfully assert that these claims are allowable over the cited prior art for at least these reasons.

2. Claims 2, 14, 41, and 48

Applicants respectfully assert that claims 2, 14, 41, and 48 are not obvious over Berkley because Berkley does not disclose, teach, or suggest all of the limitations of the rejected claims. Specifically, Berkley does not disclose, teach, or suggest the features of claim 1, as discussed in the

previous section for the rejection of claims 6, 18, 40, 44, and 47. Moreover, Berkley does not disclose, teach, or suggest "determining whether said method is non-synthetic" and "modifying said method if said method is non-synthetic," as recited in claim 2. Claims 14, 41, and 48 each contain similar features. Applicants respectfully assert that the claims are in condition for allowance.

For all the reasons discussed above, it would not be obvious to one of ordinary skill in the art to modify Berkley to include the features recited in claims 2, 14, 41, and 48. Additionally, Berkley cannot be modified to determine and modify methods based on "whether said method is non-synthetic" because all methods within an active class must be modified, regardless of the attributes of the methods. For the same reasons that the knowledge of one having ordinary skill in the art adds nothing regarding the feature of "whether said method calls another method," it would also not be obvious to modify Berkley to include the "non-synthetic" limitation using this knowledge. Therefore, Berkley would not lead one of ordinary skill in the art to develop the claimed features as recited in claims 2, 14, 41, and 48. Applicants respectfully assert that these claims are allowable over the cited prior art for at least these reasons.

3. Claims 3, 15, and 42

Applicants respectfully assert that claims 3, 15, and 42 are not obvious over Berkley because Berkley does not disclose, teach, or suggest all of the limitations of the rejected claims. Specifically, Berkley does not disclose, teach, or suggest the features of claim 1, as discussed in the previous section for the rejection of claims 6, 18, 40, 44, and 47. Moreover, Berkley does not disclose, teach, or suggest "determining whether said method has an access level of public or package" and "modifying said method if said method has an access level of public or package," as recited in claim 3. Claims 15 and 42 each contain similar features. Applicants respectfully assert that the claims are in condition for allowance.

For all the reasons discussed above, it would not be obvious to one of ordinary skill in the art to modify Berkley to include the features recited in claims 3, 15, and 42. Additionally, Berkley cannot be modified to determine and modify methods based on "whether said method has an access

level of public or package" because all methods within an active class must be modified, regardless of the attributes of the methods. For the same reasons that the knowledge of one having ordinary skill in the art adds nothing regarding the feature of "whether said method calls another method," it would also not be obvious to modify Berkley to include the "access level of public or package" limitation using this knowledge. Therefore, Berkley would not lead one of ordinary skill in the art to develop the claimed features as recited in claims 3, 15, and 42. Applicants respectfully assert that these claims are allowable over the cited prior art for at least these reasons.

4. Claims 5, 17, and 49-52

Applicants respectfully assert that claims 5, 17, and 49-52 are not obvious over Berkley because Berkley does not disclose, teach, or suggest all of the limitations of the rejected claims. Specifically, Berkley does not disclose, teach, or suggest the features of claim 1, as discussed in the previous section for the rejection of claims 6, 18, 40, 44, and 47. Moreover, Berkley does not disclose, teach, or suggest "determining whether said method is non-synthetic and has an access level of public or package" and "modifying said method if said method is non-synthetic and has an access level of public or package," as recited in claim 5. Claims 17 and 49-52 each contain similar features. Applicants respectfully assert that the claims are in condition for allowance.

For all the reasons discussed above, it would not be obvious to one of ordinary skill in the art to modify Berkley to include the features recited in claims 5, 17, and 49-52. In Berkley, all methods within an active class must be modified, regardless of the attributes of the methods, as discussed above in the arguments section for the rejection of claims 2, 14, 41, and 48 and the rejection of claims 3, 15, and 42. Therefore, Berkley would not lead one of ordinary skill in the art to develop the claimed features as recited in claims 5, 17, and 49-52. Applicants respectfully assert that these claims are allowable over the cited prior art for at least these reasons.

Based on the above, it is respectfully requested that the rejection of claims 2, 3, 5, 6, 14, 15, 17, 18, 40-42, 44, and 47-50 under 35 U.S.C. §103(a) be withdrawn. Additionally, it is respectfully

asserted that claims 51 and 52 each recite limitations of previously entered claims without adding any new matter and that these claims should be entered and allowed for at least the reasons stated in this subsection and Subsection A of the Argument Section.

D. Rejection of Claims 9, 11, 19, 21-24, 26-35, 37, 38, 45, and 46 Under 35 U.S.C. §103(a)

Claims 9, 11, 19, 21-24, 26-35, 37, 38, 45, and 46 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Berkley in view of Berry. Applicants respectfully traverse the rejection as follows.

1. Claims 9, 11, 19, 21, 22, 27, 28, 29-33, 38, 45, and 46

Applicants respectfully assert that claims 9, 11, 19, 21, 22, 27, 28, 29-33, 38, 45, and 46 are not obvious over Berkley in view of Berry because the cited prior art, alone or in combination, does not disclose, teach, or suggest all of the limitations of the rejected claims. Specifically, neither Berkley nor Berry discloses, teaches, or suggests "determining whether to modify said method, said step of determining whether to modify said method includes determining whether said method calls another method" and "modifying said method for a particular purpose if said method calls another method," as recited in claim 1. Claims 9, 11, 19, 21, 22, 27, 28, 29-33, 38, 45, and 46 each contain similar features. Applicants respectfully assert that the claims are in condition for allowance.

For all the reasons discussed above, it would not be obvious to one of ordinary skill in the art to modify Berkley to include the features recited in claims 9, 11, 19, 21, 22, 27, 28, 29-33, 38, 45, and 46 (see above argument against rejection of claims 6, 18, 40, 44, and 47). Additionally, Berry does not disclose, teach, or suggest these features. Instead, Berry discloses inserting a hook into a class when an exception is called so that the method throwing the exception can be identified, yet no determination or modification is made based on "whether said method calls another." Berry adds nothing to Berkley regarding this feature because Berry does not disclose, teach, or suggest how to modify Berkley to incorporate the feature of "modifying" particular methods based on criteria, such

as "whether said method calls another method," instead of "modifying" methods for an entire class, regardless of any attributes of the methods within the class as disclosed in Berkley. Therefore, the combination of Berkley and Berry would not lead one of ordinary skill in the art to develop the claimed features as recited in claims 9, 11, 19, 21, 22, 27, 28, 29-33, 38, 45, and 46. Applicants respectfully assert that these claims are allowable over the cited prior art for at least these reasons.

2. Claims 23 and 34

Applicants respectfully assert that claims 23 and 34 are not obvious over Berkley in view of Berry because the cited prior art, alone or in combination, does not disclose, teach, or suggest all of the limitations of the rejected claims. Specifically, neither Berkley nor Berry discloses, teaches, or suggests the features of claim 1, as discussed in the previous section for the rejection of claims 9, 11, 19, 21, 22, 27, 28, 29-33, 38, 45, and 46. Claims 23 and 34 contain similar features. Moreover, neither Berkley nor Berry discloses, teaches, or suggests "determining whether said methods are non-synthetic" and "modifying said methods if said methods are determined to be non-synthetic," as recited in claim 23. Claim 34 contains similar features. Applicants respectfully assert that the claims are in condition for allowance.

For all the reasons discussed above, it would not be obvious to modify Berkley in view of Berry to include the features recited in claims 23 and 34. Additionally, Berry adds nothing to Berkley regarding the feature of "determining" and "modifying" methods that are "non-synthetic" because Berry does not disclose, teach, or suggest this feature. One of ordinary skill in the art would not use Berry to modify Berkley to include this feature because Berry adds nothing suggesting that Berkley can be modified to include the modification of methods based on criteria (e.g. "non-synthetic") as opposed to modifying all methods within a class. Therefore, the combination of Berkley and Berry would not lead one of ordinary skill in the art to develop the claimed features as recited in claims 23 and 34. Applicants respectfully assert that these claims are allowable over the cited prior art.

3. Claims 24 and 35

Applicants respectfully assert that claims 24 and 35 are not obvious over Berkley in view of Berry because the cited prior art, alone or in combination, does not disclose, teach, or suggest all of the limitations of the rejected claims. Specifically, neither Berkley nor Berry discloses, teaches, or suggests the features of claim 1, as discussed in the previous section for the rejection of claims 9, 11, 19, 21, 22, 27, 28, 29-33, 38, 45, and 46. Claims 24 and 35 contain similar features. Moreover, neither Berkley nor Berry discloses, teaches, or suggests "determining whether said methods have an access level of public or package" and "modifying said methods if said methods are determined to have an access level of public or package," as recited in claim 24. Claim 35 contains similar features. Applicants respectfully assert that the claims are in condition for allowance.

For all the reasons discussed above, it would not be obvious to modify Berkley in view of Berry to include the features recited in claims 24 and 35. Additionally, Berry adds nothing to Berkley regarding the feature of "determining" and "modifying" methods that "have an access level of public or package" because Berry does not disclose, teach, or suggest this feature. One of ordinary skill in the art would not use Berry to modify Berkley to include this feature because Berry adds nothing suggesting that Berkley can be modified to include the modification of methods based on criteria (e.g. "access level of public or package") as opposed to modifying all methods within a class. Therefore, the combination of Berkley and Berry would not lead one of ordinary skill in the art to develop the claimed features as recited in claims 24 and 35. Applicants respectfully assert that these claims are allowable over the cited prior art

4. Claims 26 and 37

Applicants respectfully assert that claims 26 and 37 are not obvious over Berkley in view of Berry because the cited prior art, alone or in combination, does not disclose, teach, or suggest all of the limitations of the rejected claims. Specifically, neither Berkley nor Berry discloses, teaches, or suggests the features of claim 1, as discussed in the previous section for the rejection of claims 9, 11, 19, 21, 22, 27, 28, 29-33, 38, 45, and 46. Claims 26 and 37 contain similar features. Moreover,

neither Berkley nor Berry discloses, teaches, or suggests "determining whether said methods are non-synthetic and have an access level of public or package" and "modifying said methods if said methods are determined to be non-synthetic and have an access level of public or package," as recited in claim 26. Claim 37 contains similar features. Applicants respectfully assert that the claims are in condition for allowance.

For all the reasons discussed above, it would not be obvious to one of ordinary skill in the art to use Berry to modify Berkley to include the features recited in claims 26 and 37. In Berkley, all methods within an active class must be modified, regardless of the attributes of the methods. Additionally, Berry adds nothing regarding the features recited in claims 26 and 37 because Berry adds nothing to suggest modifying Berkley to include the modification of methods based on criteria (e.g. "non-synthetic" or "access level or public or package") as opposed to modifying all methods within a class. Therefore, Berkley would not lead one of ordinary skill in the art to develop the claimed features as recited in claims 26 and 37. Applicants respectfully assert that these claims are allowable over the cited prior art for at least these reasons.

Based on the above, it is respectfully requested that the rejection of claims 9, 11, 19, 21-24, 26-35, 37, 38, 45, and 46 under 35 U.S.C. §103(a) be withdrawn.

E. CONCLUSION

Based on the above, it is respectfully submitted that claims 1-3, 5-15, 17-24, 26-35, 37-42, and 44-50 are patentable over the cited references, and it is respectfully requested that the rejections of these claims be withdrawn. Additionally, it is respectfully submitted that claims 51 and 52 each recite limitations of previously entered claims and that these claims should be entered and allowed based on the above arguments.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this Appeal Brief, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: March 10, 2008

By: /Michelle Esteban/
Michelle Esteban
Reg. No. 59,880

VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105
Telephone: (415) 369-9660
Facsimile: (415) 369-9665

VIII. CLAIMS APPENDIX *(37 C.F.R. §41.37(c)(viii))*

1. (previously presented) A process for monitoring, comprising:

accessing a method;

determining whether to modify said method, said step of determining whether to modify said method includes determining whether said method calls another method ; and

modifying said method for a particular purpose if said method calls another method .

2. (previously presented) A process according to claim 1, wherein:

said step of determining whether to modify said method includes determining whether said method is non-synthetic; and

said step of modifying includes modifying said method if said method is non-synthetic and said method calls another method.

3. (previously presented) A process according to claim 1, wherein:

said step of determining whether to modify said method includes determining whether said method has an access level of public or package; and

said step of modifying includes modifying said method if said method has said access level of public or package and said method calls another method.

4. (cancelled)

5. (previously presented) A process according to claim 1, wherein:

said step of determining whether to modify said method includes determining whether said method is non-synthetic and has an access level of public or package; and

said step of modifying includes modifying said method if said method is non-synthetic, has said access level of public or package, and said method calls another method.

6. (previously presented) A process according to claim 1, wherein:

said step of determining whether to modify said method includes determining whether said method can be called by a sufficient scope of one or more other methods; and

said step of modifying said method includes modifying said method if said method can be called by said sufficient scope of one or more other methods and said method calls another method.

7. (original) A process according to claim 1, wherein:

said step of modifying includes modifying object code.

8. (original) A process according to claim 1, wherein:

said step of modifying includes adding a tracer for said method.

9. (original) A process according to claim 1, wherein:

said step of modifying includes adding a timer for said method.

10. (original) A process according to claim 1, wherein:

said step of modifying includes adding exit code and start code to existing object code.

11. (original) A process according to claim 10, wherein:

said start code starts a tracing process;

said exit code stops said tracing process;

said exit code is positioned to be executed subsequent to original object code;

said step of adding exit code includes adding an instruction to jump to said exit code from said original object code;

said step of adding exit code includes adding an entry in an exceptions table; and

said step of adding an entry in said exceptions table includes adding a new entry into said exceptions table for said method, said new entry indicates a range of indices corresponding to said original object code, said new entry includes a reference to said exit code and said new entry indicates that said new entry pertains to all types of exceptions.

12. (original) A process according to claim 1, wherein:

said particular purpose is to add a first tracer.

13. (previously presented) A process for monitoring, comprising:

determining which methods of a set of methods call one or more other methods ; and

using a first tracing mechanism for said methods determined to call one or more other methods without using said first tracing mechanism for methods not determined to call one or more other methods

14. (previously presented) A process according to claim 13, wherein:

said step of determining includes determining whether said methods are non-synthetic; and

said step of using includes using said first tracing mechanism if said methods are determined to be non-synthetic and said methods call one or more other methods.

15. (previously presented) A process according to claim 13, wherein:

said step of determining includes determining whether said methods have an access level of public or package; and

said step of using includes using said first tracing mechanism if said methods are determined to have said access level of public or package and said methods call one or more other methods.

16. (cancelled)

17. (previously presented) A process according to claim 13, wherein:

said step of determining includes determining whether said methods are non-synthetic and have an access level of public or package; and

said step of using includes using said first tracing mechanism if said methods are non-

synthetic, have said access level of public or package, and said methods call one or more other methods.

18. (previously presented) A process according to claim 13, wherein:

said step of determining includes determining whether said methods can be called by a sufficient scope of one or more other methods; and

said step of using includes using said first tracing mechanism if said methods can be called by said sufficient scope of one or more other methods and said methods call one or more other methods.

19. (original) A process according to claim 13, wherein:

said step of using a first tracing mechanism includes adding and using timers for said methods.

20. (original) A process according to claim 13, wherein:

said step of using a first tracing mechanism includes modifying existing object code to add said first tracing mechanism.

21. (original) A process according to claim 20, wherein:

said first tracing mechanism includes timers for said methods.

22. (previously presented) One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a process comprising:

determining which methods of a set of methods to modify, said step of determining includes determining whether said methods call one or more other methods ; and

modifying for a particular purpose those methods that are determined to call one or more other methods .

23. (previously presented) One or more processor readable storage devices according to claim 22, wherein:

said step of determining includes determining whether said methods are non-synthetic; and

said step of modifying includes modifying said methods if said methods are determined to be non-synthetic and said methods call one or more other methods.

24. (previously presented) One or more processor readable storage devices according to claim 22, wherein:

said step of determining includes determining whether said methods have an access level of public or package; and

said step of modifying includes modifying said methods determined to have said access level of public or package and said methods call one or more other methods.

25. (cancelled)

26. (previously presented) One or more processor readable storage devices according to claim 22, wherein:

said step of determining includes determining whether said methods are non-synthetic and have an access level of public or package; and

said step of modifying includes modifying said methods if said methods are determined to be non-synthetic, have said access level of public or package, and said methods call one or more other methods.

27. (previously presented) One or more processor readable storage devices according to claim 22, wherein:

said step of determining includes determining whether said methods can be called by a sufficient scope of one or more other methods; and

said step of modifying includes modifying said methods if said methods can be called by said sufficient scope of one or more other methods and said methods call one or more other methods.

28. (original) One or more processor readable storage devices according to claim 22, wherein:

said step of modifying includes modifying existing object code.

29. (original) One or more processor readable storage devices according to claim 22, wherein:

said step of modifying includes adding tracers.

30. (original) One or more processor readable storage devices according to claim 22, wherein:

said step of modifying includes adding timers.

31. (original) One or more processor readable storage devices according to claim 22, wherein:

said step of modifying includes adding exit code and start code to existing object code.

32. (original) One or more processor readable storage devices according to claim 31, wherein:

said start code starts a tracing process;

said exit code stops said tracing process;

said exit code is positioned to be executed subsequent to original object code;

said step of adding exit code includes adding an instruction to jump to said exit code from said original object code;

said step of adding exit code includes adding an entry in an exceptions table; and

said step of adding an entry in said exceptions table includes adding a new entry into said

exceptions table for said method, said new entry indicates a range of indices corresponding to said original object code, said new entry includes a reference to said exit code and said new entry indicates that said new entry pertains to all types of exceptions.

33. (previously presented) One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a process comprising:

determining whether to trace a method, said step of determining includes determining whether said method calls another method ; and

tracing said method for a particular purpose if said method calls another method .

34. (previously presented) One or more processor readable storage devices according to claim 33, wherein:

said step of determining includes determining whether said method is non-synthetic; and

said step of tracing includes tracing said method if said method is determined to be non-synthetic and said method calls another method.

35. (previously presented) One or more processor readable storage devices according to claim 33, wherein:

said step of determining includes determining whether said method has an access level of public or package; and

said step of tracing includes tracing said method if said method is determined to have said access level of public or package and said method calls another method.

36. (cancelled)

37. (previously presented) One or more processor readable storage devices according to claim 33, wherein:

said step of determining includes determining whether said method is non-synthetic and has an access level of public or package; and

said step of tracing includes tracing said method if said method is determined to be non-synthetic, have said access level of public or package, and said method calls another method.

38. (previously presented) One or more processor readable storage devices according to claim 33, wherein:

said step of tracing includes timing said method.

39. (previously presented) An apparatus capable of monitoring, comprising:

means for determining whether a method calls another method ; and

means for tracing said method for a particular purpose only if said method calls another method .

40. (previously presented) An apparatus capable of monitoring, comprising:

a storage device; and

one or more processors in communication with said storage device, said one or more processors perform a process comprising:

accessing a method,

determining whether said method calls one or more different methods and can be called by a sufficient scope of one or more other methods, and

tracing said method for a particular purpose if said method calls one or more different methods and can be called by a sufficient scope of one or more other methods.

41. (previously presented) An apparatus according to claim 40, wherein:

said step of determining includes determining whether said method is non-synthetic; and

said step of tracing includes tracing said method if said method is determined to be non-synthetic and said method calls one or more different methods.

42. (previously presented) An apparatus according to claim 40, wherein:

said step of determining includes determining whether said method has an access level of public or package; and

said step of tracing includes tracing said method if said method is determined to have said access level of public or package and said method calls one or more different methods.

43. (cancelled)

44. (original) An apparatus according to claim 40, wherein:

said process further includes modifying existing object code for said method in order to add a first tracing mechanism.

45. (original) An apparatus according to claim 44, wherein:

said first tracing mechanism includes a timer.

46. (original) An apparatus according to claim 40, wherein:

said step of tracing includes timing said method.

47. (previously presented) A process for monitoring, comprising:

accessing a method;

determining whether said method is complex, said step of determining includes determining that said method is complex if said method calls another method; and

modifying said method for a particular purpose only if said method is determined to be complex.

48. (previously presented) A process according to claim 47, wherein:

said step of determining includes determining that said method is complex if said method is

non-synthetic and said method calls another method.

49. (previously presented) A process according to claim 47, wherein:

said step of determining includes determining that said method is complex if said method has an access level of public or package, said method is non-synthetic, and said method calls another method.

50. (previously presented) A process according to claim 49, wherein:

said step of modifying includes adding a tracer for said method.

51. (previously presented) A process according to claim 5, wherein:

said step of modifying includes adding a tracer for said method.

52. (previously presented) An apparatus according to claim 40, wherein:

said step of determining includes determining whether said method is non-synthetic and whether said method has an access level of public or package; and

said step of tracing includes tracing said method if said method is determined to be non-synthetic, said method is determined to have an access level of public or package, and said method calls one or more different methods.

IX. EVIDENCE APPENDIX *(37 C.F.R. §41.37(c)(ix))*

None

X. RELATED PROCEEDINGS APPENDIX *(37 C.F.R. §41.37(c)(x))*

None